SE 6356 Software Maintenance, Evolution
and Re-Engineering - Fall 2016
Assignment #1 - Due date: 10/04/2015
-- part 2 --

Introduction

In this second part of assignment 1, you will be implementing three change requests for fEMR, a Web-based Electronic Medical Record. Instructions for setting up the system are given below.

Installing IntelliJ

The IDE used to develop and maintain fEMR is IntelliJ Ultimate. Request a copy of IntelliJ Ultimate and install it. You will need a license to use the Ultimate version; however, Jetbrains, the company that owns the IDE, offers free licenses to students. Apply for a free license here.

Creating fEMR's database

The following are the instructions to create fEMR's database (DB) from a MySQL dump, which contains anonymized data from real patients:

1. Install MySQL server 5.6
2. Create a user with administrative privileges called `femruser`
3. Log into MySQL with the created user and create a DB schema for fEMR:
   ```
   mysql -u femruser -p -e "CREATE DATABASE femrdb"
   ```
4. Import this MySQL dump into the created schema `femrdb`:
   ```
   mysql -u femruser -p femrdb < 1-9-2016_noNames.sql
   ```

Setting up and running fEMR

In fEMR's contributing webpage you will find detailed instructions on how to install, configure, and run fEMR. As a summary, you will have to do the following:

1. Install the required software for development: JDK 8, Play Framework 2.3.7, last version of Git.
2. Fork fEMR's repository.
3. Clone the forked repository in your machine.
4. Clean and compile the code using the `activator` utility from the Play framework.
5. Import and configure the project in IntelliJ. In particular, change the connection parameters of the DB in the file application.dev.conf
6. Run fEMR using IntelliJ. The first run of the application will show a screen with the title "Database 'default' needs evolution!". Click on the button "Apply this script now" to update the database to its last version.
7. Log into the application using the credentials below. If prompted, change the password.
   Username: anonymoususer@gmail.com
   Password: Password1
8. Once logged in, you should be able to use the system. For example, go to the Admin/Users web page to list the users currently registered in the system.

Before implementing the change requests…

The contributing webpage of fEMR provides details on how to prepare your development environment before implementing each change request. In particular, there are two major steps you should follow:

1. Synchronize the master branch of your forked repository with the master branch of fEMR's repository, to ensure you repository is always up-to-date with fEMR's latest changes.
2. Create a development branch for you change request. To implement a change request, use the created branch as opposed to the master branch. All the change requests that you will implement are "improvements", so refer to the git commands for improvements in the contributing webpage.

Change requests

Read and understand the assigned change requests. In case of any question about them, consult the TA or fEMR team members via Slack or e-mail.

1. FEMR-158: don't require the user editing a user to fill out the "Change User Password" input fields
2. FEMR-208: encounter PDF not displaying amount of prescription dispensed
3. FEMR-137: flag birthdays as being accurate or a guess

Deliverables

1. Modified source code of the system. Push your changes into your forked repository. Your repository should contain 3 branches, one for each change request. Add the TA and instructor as collaborators of your repository (GitHub accounts: ojcchar and amarcusgit)
2. Change log for each change request. Fill-in the attached change log for each implemented change request. Add it to your GitHub repository. Submit the logs using eLearning.

Notes

- You will not receive credit for this assignment if:
    - You do not fill the logs.
    - Your code does not compile.
    - You did not create a branch for each change request.
    - You do not add the TA and instructor as collaborator in your repository.
- The logs should not contain gaps in the process you followed to implement the changes. Provide enough details about queries, tools used, classes visited, decisions, etc. Any person should be able to precisely understand the process you followed. It is very important to clearly describe the rationale of your tasks and decisions.
- Do not forget to evaluate the process you went through and provide enough details in the conclusions in the change request logs. Include details on why some steps were harder than others.
- Specify in the logs who of the team members did the coding and who did the documentation.
- Proofread carefully your logs.